EECS 3201 Digital Logic Laboratory

Lab3: Addition and Subtraction Calculator

Objective

The objective of this lab is to implement a simple 4-bit calculator for addition and subtraction, using the seven segment displays on the DE10-Lite to display both the arguments and result. You will gain further experience with arithmetic in digital logic.

Reference Material

Altera DE10 manual and resources, available from the course web site.

You will use several instances of your seven-segment decoder module from Lab 2. If this is not possible for some reason (e.g. you couldn't get it to work), you can use a module from somewhere else (e.g., a classmate, or the internet), **as long as you give appropriate credit**.

Task

Adder and Subtractor

Implement a four-bit adder in Verilog. You might do it as follows:

- Create a module that implements a full adder.
- Using four full adders, connected as described in the lecture, implement a four-bit ripple-carry adder.
- Modify your four-bit adder with an additional input bit, called s, so that if s = 0, the adder implements addition, and if s = 1, the adder implements 2's complement subtraction.

You might also do it with procedural Verilog, i.e. performing the calculations within an "always" block. Either way is correct and worth full marks, as is any other method that produces the result specified below.

Inputs and Outputs

Whichever way you implement the circuit, the inputs and outputs to your module should work in the following way:

- Inputs: Two (unsigned) four-bit binary numbers, a0 and a1; and the subtraction control bit s

- Outputs: One four-bit binary number as the result (a0+a1 for addition, and a0-a1 for subtraction); and the carry out bit "cout" from the final full adder

Inputs a0, a1, and s should be assigned to switches on the DE10-Lite (which ones are up to you).

For outputs, you should use the seven-segment displays to display the digit corresponding to both inputs (a0 and a1), as well as the result. Use HEX5 (the leftmost display) for a0, HEX3 for a1, and HEX0 (the rightmost display) for the result.

To do this, you will need to use three seven-segment decoder modules: one for a0, one for a1, and one for the result. You can use the one you wrote in Lab 2, or you can use one from a friend or from the internet. (Important note: if you didn't write your seven-segment decoder yourself, **you must credit the source**; do so in a comment in Verilog. Failure to credit your sources is an academic honesty violation.)

Example outputs are given below.

What about the carry out (cout)?

The output cout means different things in addition and subtraction:

- When doing addition, cout = 1 if the result is greater than hexadecimal F, so that there is a 1 in the position of the next digit. That is, if cout = 1, then the result is in 10, 11, 12, …, 1F. On the other hand, cout = 0 means that the result is completely expressed in the four bits of the output, i.e., the result is in 0, 1, 2, …, F. You can check this with an example, e.g. try adding 8+9 in binary and see where the last carry goes.
- When doing subtraction, cout = 1 if the result is positive and cout = 0 if the result is negative. This is natural if you use the 2's complement method: in this case, the four-bit output of the adder expresses the result in two's complement form, i.e., in hexadecimal: F = -1, E = -2, and so on. If you don't use the 2's complement method, it might not work this way, but cout should still indicate the sign of the subtraction: 1 if positive, and 0 if negative.

Modify your output as follows, using the HEX1 seven-segment display (the display to the left of HEX0, which displays the digit corresponding to the four-bit output of the adder).

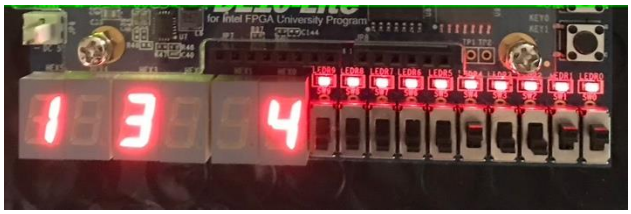
- For addition (*):
    - If cout = 1, a 1 should appear on HEX1.
    - If cout = 0, HEX1 should be blank.
- For subtraction (**):
    - If cout = 1, HEX1 should be blank.
    - If cout = 0, HEX1 should contain a minus sign (only the middle LED illuminated), and the digit on HEX0 should display the magnitude of the result. (If you used the 2's complement method, the display should undo

the two's complement. For example, if the result of the subtraction is E, which is -2 in two's complement, then HEX0 should display 2, not E; together, HEX1 and HEX0 should read "-2".)

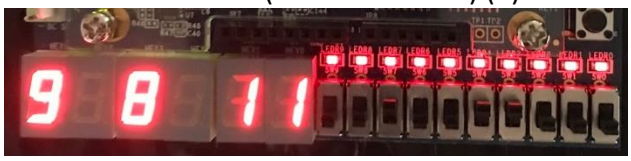
When implementing this part, describe your approach in Verilog comments.

Examples

Addition: 1 + 3 = 4



Addition: 9 + 8 = 11 (in hexadecimal) (*)



Subtraction: 8 – 6 = 2

Subtraction: 5 – 7 = -2 (**)

Subtraction: E – F = -1 (**)

The implemented circuits must be demonstrated to the TA who will note a completed lab and ask questions about your design. When implementing the circuit be sure to use the switches and lights to make it easy to demonstrate your circuits.

Evaluation

Lab demonstration, in-lab explanations and answers, debug and test approach. [2 marks for fully working]

Lab report containing [graded out of 50, weight 2 marks of the overall lab grade]:

- Explanation of design approach and design documentation such as truth tables, maps, Boolean expressions and other design aids.
- Fully documented Verilog source (submit as .v files)
- Test patterns and/or a testing strategy

EECS3201 Lab

CRITERIA	SCALES			
	Unacceptable	Marginal	Good	Excellent
Demonstration marked in lab, added to this score	0.00 Did not work, no reasonable explanation or understanding of why.	0.00 Demonstration of some of the functionality and some ability to explain how and why the circuit functions.	0.00 Demonstration of most or all of the functionality. Ability to explain their design but limited ability to compare with other approaches or justiy decisions.	0.00 The circuit works fully and design decisions can be explained and justified.
Code	0.00 No or unsuitable code.	7.00 Some suitable code is presented but it is incorrect, poorly documented or incomplete.	13.00 Code for all or the majority of the function is provided. Some errors or issues in documentation, organization or design.	20.00 Well designed, documented and organized code is provided for all functionality.
Test Strategy	0.00 No or unsuitable strategy.	7.00 Some testing and/or strategy for portions of the code provided but it is incomplete or not systematic.	13.00 Valid strategy for most of the functionality is described and implemented. Some issues with effectiveness, efficiency or errors in strategy or implementation.	20.00 Well designed, efficient and effective test and simulation strategy is described and implemented.
Format	0.00 Unsuitable, did not follow directions.	4.00 All components present but poorly written, poor quality figures or organizational issues.	7.00 Complete and formatted to specifications but some issues with writing, organization or documentation. Repetitive, overly long or missing minor components.	10.00 Well documented and complete. Descriptions are brief but complete and effective.